Speed up your calibration with CAN FD

Dr. David Gary Hickman, Etas

A description of the new CAN FD format is given, describing all the changes, not only for the data rate and payload size. An analysis of how this affects the overall data rate of the CAN bus is shown. The changes in the new measure and calibration standard XCP1.2 to support CAN FD are described, and then an analysis of the busload can be calculated with CAN FD, plus the likely improvements in the XCP download and flash times. Benefits can also be made to the measure performance using DAQ. The CAN FD changes mean either the same measure data can be sent with reduced bandwidth, more data can be send at the same rasters, or data can be send in faster raster then previously possible.

Getting more performance from the CAN system was not easy because the CAN speed is limited to ensure the message arbitration occurs reliably. In addition the message structure means each message has a large overhead compared to payload. The solution is CAN FD, which sends the data part of the message faster, and increases the payload size from 8 to 64 bytes. In this paper we will look at the impact this can have on measurement and calibration using XCP on CAN FD.

Main changes with CAN FD

The speed the CAN bus can operate at is determined by the need to complete the message arbitration]. So the speed is dependent on the length and topology of the bus. For passenger vehicles the CAN speed is between 500 Kbit/s - 1Mbit/s, while in commercial vehicles, bus speeds of 250 kbit/s are more typical.
With CAN FD [1], the speed of the arbitration remains the same. However at the end of arbitration when only one node is sending, the bus speed can be increased. This is the idea of CAN FD. The payload part of the CAN message, and the CRC (Cyclic Redundancy Check) are send at a higher bit rate then the arbitration.

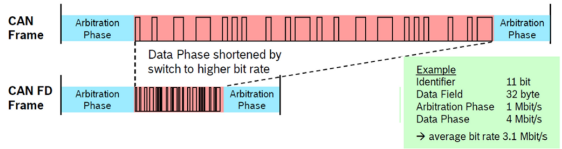

Figure 1: Effect of bitrate change

Figure 1 shows the effect of the bitrate change. In the example the flexible data rate is set to 4Mbits/s. The effect is to reduce the required time to send the data, allowing more CAN frames to be sent on the bus.
In addition, CAN FD allows the payload size to be increased. Before, the payload was limited to 8 bytes. With CAN FD this can be increased to a maximum of 64 bytes.
The same DLC code format is used as for legacy CAN. With CAN FD, 7 new payload sizes are made available representing the full use of the DLC_3 bit as shown in Figure 2.

	Number of Data Bytes	Data Length Code			
		DLC_3	DLC_2	DLC_1	DLC_0
Codes in CAN and CAN FD Format	0	0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
CAN Format	8	1	0/1	0/1	0/1
Codes in CAN FD Format	8	1	0	0	0
	12	1	0	0	1
	16	1	0	1	0
	20	1	0	1	1
	24	1	1	0	0
	32	1	1	0	1
	48	1	1	1	0
	64	1	1	1	1

Figure 2: Payload Data Length Codes

To support the longer payload, the Cyclic Redundancy Check (CRC) used to checksum the payload, also needs to be extended.

CAN 0-8 Bytes	•	CRC_15 =	0xC599 $(x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1)$ $(x+1)\cdot(x^7+x^3+1)\cdot(x^7+x^3+x^2+x+1)$
CAN FD 0-16 Bytes	•	CRC_17 =	0x3685B $(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^6+x^4+x^3+x^1+1)$ $(x+1)\cdot(x^{16}+x^{13}+x^{10}+x^9+x^8+x^7+x^6+x^3+1)$
CAN FD 17-64 Bytes	•	CRC_21 =	0x302899 $(x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1)$ $(x+1)\cdot(x^{10}+x^3+1)\cdot(x^{10}+x^3+x^2+x^1+1)$

Figure 3: CAN FD CRC codes

Figure 3 shows the new CRC codes available. For the CAN FD messages up to 16 bytes we use a 17 stage polynomial. Between 17 and 64 byte payloads a 21 stage polynomial is used.
To identify CAN FD messages, the CAN FD frames have a new message structure shown in Figure 4.

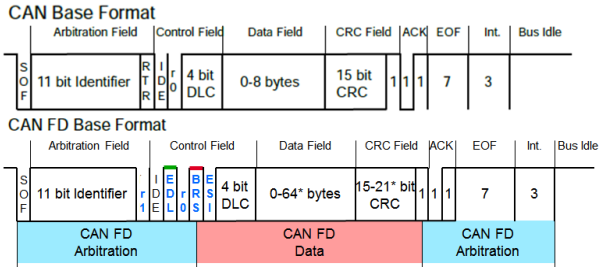

Figure 4: Comparison of CAN and CAN FD messages in base frame formats (11 bit identifier)

After the arbitration, a CAN FD frame will always send a dominant r1 (reserved) frame instead of the RTR (Remote Transmission Request). This means CAN FD does not support remote frames. The recessive EDL (Extended Data Length) is the identifier of a CAN FD frame. The BRS (Bit Rate Switch) is transmitted recessive and is used as the switch to change the bit rate to the flexible data rate.
A consequence of the different frame structure is that CAN and CAN FD messages cannot be easily mixed on a single bus. The reason legacy CAN nodes will detect an error when it receives a CAN FD frame

Other changes with CAN FD

There are some additional changes with CAN FD which are not directly related to increased payload size and flexible data rate.
The ESI (Error State Indicator) is a new bit for CAN FD frames. It allows error passive and error active nodes to be identified to help find the cause of bus failures.
In addition the stuff bit method for the CRC sequence is changed. For CAN FD, stuff bits will be inserted at fixed positions, even if the preceding bits do not satisfy the bit stuffing criteria. So a stuff bit will be inserted in the first bit of the CRC sequence, then after each fourth bit of the CRC sequence. The inserted stuff bit will be the inverse of the preceding bit.

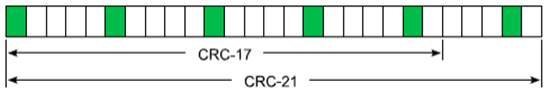

Figure 5: CRC sequence Stuff Bits

This means the CRC-17 always has 5 stuff bits, and the CRC-21 always 6 stuff bits.

Data bit rates for CAN FD

Calculating the data bit rates for CAN FD is complicated by the fact there are 2 data rates, and the messages can have variable payloads. In addition, because of the bit stuffing, the message lengths can change even if the payload remains constant

Frame Type	Payload [Bytes]	Frame length Arb.	Data	Whole	Comments
Legacy	8	28	83	111	standard / no stuff bits
FD	8	29	91	120	standard / no stuff bits
Legacy	8	31	103	134	standard / max stuff bits
FD	8	32	108	140	standard / max stuff bits
Legacy	8	47	83	130	extended / no stuff bits
FD	8	48	91	139	extended / no stuff bits
Legacy	8	53	103	156	extended / max stuff bits
FD	8	54	108	162	extended / max stuff bits
FD	64	29	544	573	standard / no stuff bits
FD	64	32	673	705	standard / max stuff bits
FD	64	48	544	592	extended / no stuff bits
FD	64	54	673	727	extended / max stuff bits

Figure 6: Max and Min CAN frame lengths

Figure 6 shows the max and min sizes of the frame length with no and maximum bit stuffing. For the bit rate calculations, we will assume no bit stuffing. In addition, for all the calculations we will assume an arbitration rate of 1 Mbit/s. The 1 Mbit/s will be used for all subsequent examples presented in this paper.
There are 2 ways of showing the data rate. One is the net bit rate, which is the rate sending the complete CAN frame data, including the arbitration and CRC information. The other is the average bit rate. This is calculated by taking only the payload part, and dividing this by the time taken to send the complete message.. This shows the rate the useful information is transferred at.
This data is plotted on Figure 7, for standard arbitration length frames with

payloads of 8, 24 and 64 bytes. What can be seen is there is a big difference between the overall and effective data rates for 8 byte payloads. This is because the ratio of payload to overall message length (message efficiency) of these low payloads is not high. The higher message efficiency of the 64 byte payload means the difference in the overall and effective data rates are much smaller

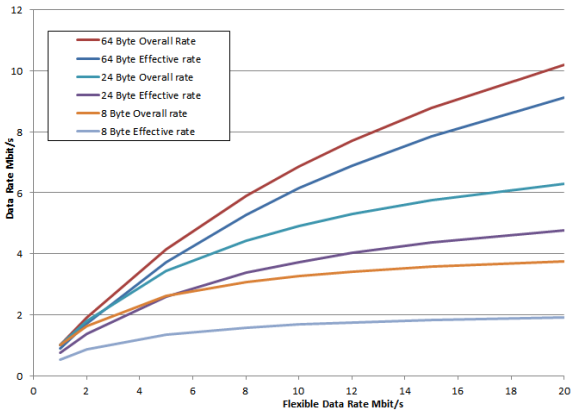

Figure 7: Net and average bit rates for different CAN loads

It is also useful to look at the effect of the bit stuffing on the data rate as this is important for analyzing the XCP measure performance.

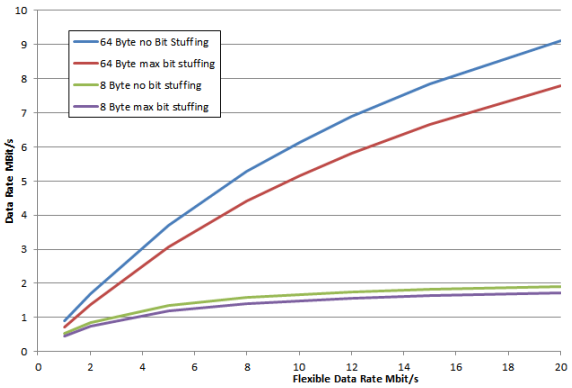

Figure 8: Effect of bit stuffing of effective data rate

Figure 8 shows the effect of bit stuffing on the effective data rate. The effect for 8 byte payloads is low. For 64 byte payloads the effect is to reduce the data rate by over 12%. This is because of the large potential number of stuff bits in the longer 64 byte payload.

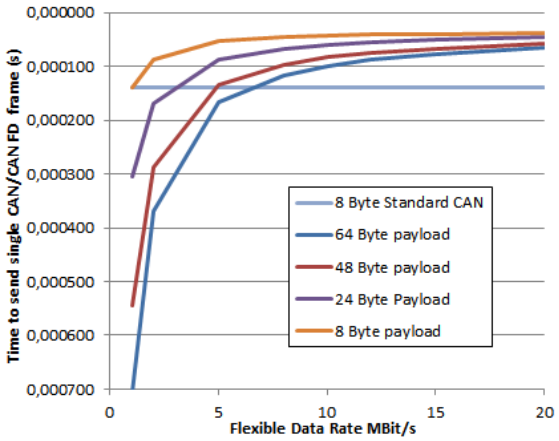

Figure 9: Time to send single legacy CAN/CAN FD from with maximum bit stuffing

Figure 9 shows the time required to send a single legacy CAN and CAN FD frame. The calculated time represents the worst case scenario, with each frame having the maximum stuff bits. The time for the 8 byte legacy CAN is plotted against all FD data rates in order to compare the send times for CAN FD and legacy CAN. CAN FD frames with 8 byte payloads are transmitted faster than legacy CAN frames. This has implications for XCP DAQ measurement because it means we can send data at faster rates. Although 64 byte CAN FD messages are longer than legacy CAN messages, when the FD is greater than 6 Mbit/s, even these are sent faster than legacy CAN messages.

XCP 1.2 features relating to CAN FD

The XCP 1.2 standard was released in June 2013 [2]. Included in the specification is the support of CAN FD. The main change is to extend the XCP on CAN Transport Layer part to additionally include CAN FD.
Part of the change in the XCP1.2 standard is to introduce a Dynamic DAQ raster check mechanism. The check itself is entirely offline and performed by the calibration tool, and relies on entries in the XCP on CAN part of the a2l file. The check itself has 3 parts: CPU usage and memory limits for building and sending the DAQ lists, and a busload limit for the DAQ massages that are send on the bus. For CAN FD we are only concerned with the busload limit. Of course, if the extra

bandwidth of CAN FD is used to send more measure data there will be an effect on the CPU and memory consumption.
The whole topic of busload and actual data rates with CAN FD is complicated and will be looked at in the next section. For the calculation of CAN FD busload, an assumption of the frame sizes must be made, based on the DLC used for the DAQ lists. The values are shown in Figure 10.

Bus	DLC	Used frame length for Calculation (Std. CAN-ID)	Used frame length for Calculation (Ext. CAN-ID)
CAN	8	120	140
CAN-FD	8	130	150
	12	170	195
	16	210	230
	20	245	265
	24	280	300
	32	320	340
	48	495	515
	64	640	660

Figure 10: CAN Frame length for raster check

The value of DLC used depends on the MAX_DLC_REQUIRED declaration in the a2l file. Because the CAN frame is not fixed due to the amount of bit stuffing, the values used in the tables represent the average CAN frame length.

In addition, the arbitration length of the standard and extended CAN IDs also need to be known. These are sent at the arbitration CAN data rate, so the calculation of the overall rate must be separated to include the arbitration and data phases.

- For standard ID arbitration is set to 30bit
- For extended ID arbitration is set to 50bit

The overall busload is calculated factored to the arbitration bit rate. So we first calculate the number of bits sent for each CAN message factored to the ratio arbitration data rate to the CAN FD data rate.

$$\#\ Bits = \#\ Arb\ Bits + \frac{\#\ Databits * BAUDRATE}{CAN\ FD\ BAUDRATE}$$

We sum this for each ODT in an Event (raster) and then sum again for every Event (or raster) selected. This total busload is divided by the arbitration data rate to calculate the bus load.

The remaining changes for the XCP specification in respect to CAN FD refer to the a2l file changes. These are listed in table 1.

Table 1: XCP1.2 a2l file changes for CAN FD

Entry	Block	Comment
MAX_DLC	CAN FD	Set the maximum Message size
CAN_FD_DATA_TRANSFER_BAUDRATE		FD Baudrate
SAMPLE_PONIT	CAN FD	Added to because new parameters required for FD baudrate part
SAMPLE_RATE	CAN FD	See above
BTL_CYCLES	CAN FD	See above
SJW	CAN FD	See above
SYNC_EDGE	CAN FD	See above
MAX_DLC_REQUIRED	CAN FD	See above
SECONDARY_SAMPLE_POINT	CAN FD	See above
TRANCEIVER_DELAY_COMPENSATION	CAN FD	Should not be required as the transceiver themselves will calculated this.

SAMPLE_RATE is the flexible data rate. The entries for SAMPLE_POINT; BLT_CYCLES and SJW are required because the flexible data rate part can have different values to the arbitration data rate, because the transmission speed is increased. The TRANSCEIVER_DELAY_COMPENSATION refers to the delay required to correctly read the sent bits. If no error is set, then the bit error detection will not function, resulting false errors being detected. Although specified in the XCP standard, actually this is not needed as the transceiver itself should determine the required delay compensation.

CAN FD effect on Data Download

One of the main interests of CAN FD up to now has been to use its increased data rate and payload size to improve the ECU data download and flash times**.** An illustration of the improvements is shown in Figure 11. This shows the time to download 500Kbytes of data using block

transfer with a block size of 256Kbytes and a message separation time (STmin) of 0sec.

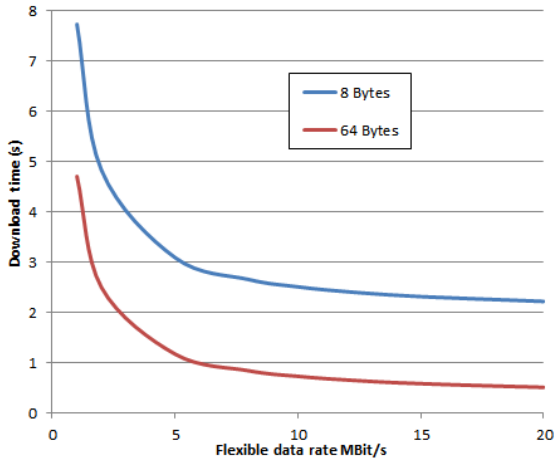

Figure 11: Data download times

The data shows significant improvements to download times can be realized, even when 8 byte payloads are used. So increasing the FD to 5 Mbits/s results in over a halving of the download time. Further improvements are made by increasing the payload.

CAN FD effect on DAQ Measurement

DAQ (**D**ata **Acq**uisition) is the method used by XCP to collect measure data. Data is collected and sent by the slave device at fixed time or event intervals. For ECUs the intervals normally correspond to internal calculation cycles. This allows signals to be measured in the same cycle the signal is generated or when it is used in a software function.

The analysis concentrates on DAQ messages having 8 and 64 byte payloads. 8 bytes represents the easiest change to CAN FD by only changing the data rate. 64 byte payload offers the best data transfer performance, and also offers advantages of better data consistency. By packing more data into a single CAN frame, there is less risk of the data being updated from the ECU controller before all the data has been sent. This is a risk if data has to be packed over several frames and the data has to be buffered longer before it is sent.

There are a number of possible ways CAN FD can affect DAQ. The first is to examine the effect of changing the CAN FD settings which keeping the same busload limit.

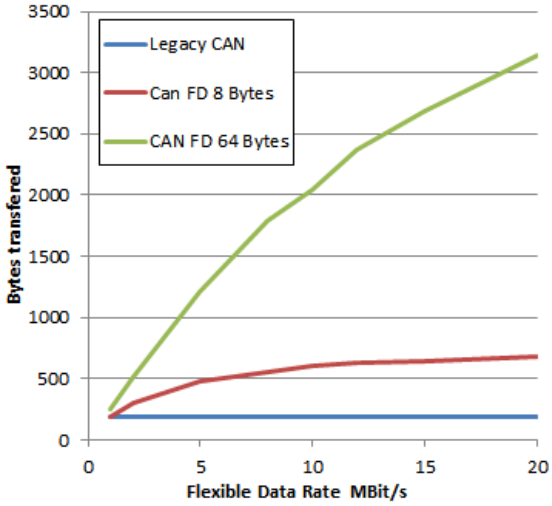

Figure 12: bytes transferred maintaining 30 % busload

Figure 12 displays the amount of measure data that can be sent if the busload is maintained constant and the FD rate and payload is increased. In this case the busload is kept to 30 %. To keep the calculation simple, we are just transferring data in a single 10ms event. Our baseline is a 30% busload with legacy 8 byte CAN, which allows us to send 184 bytes every 10ms. The graph shows how many bytes of data at 10ms represents with different CAN FD parameters and a 30% busload. The data transferred with legacy 8 byte CAN is also plotted as a comparison. Of course the legacy CAN values for the different FD rates cannot exist. We plot them in this way only to see the benefits of CAN FD:

The graph shows that maintaining an 8 byte payload and increasing the flexible data rate results in a significant increase in data transmitted. With an FD of 5 Mbit/s the data that can be sent is more than doubled from 184 to 480 bytes. If the DLC is increased to 64 bytes, then the increase in data transferred is much greater. An FD of 5 Mbit/s means an increase from 184 bytes for legacy 8 byte CAN to 1216 bytes, a 6 fold increase.

This example is important if busload is the limiting factor for XCP measure performance. This is often the case. It shows the increase in bandwidth with CAN FD can bring significant improvements to measure performance.

The next example looks at the effect on the busload when we send the same amount of data with different CAN FD settings. So again the starting point is a 30% busload with legacy CAN, sending all the data every 10ms.

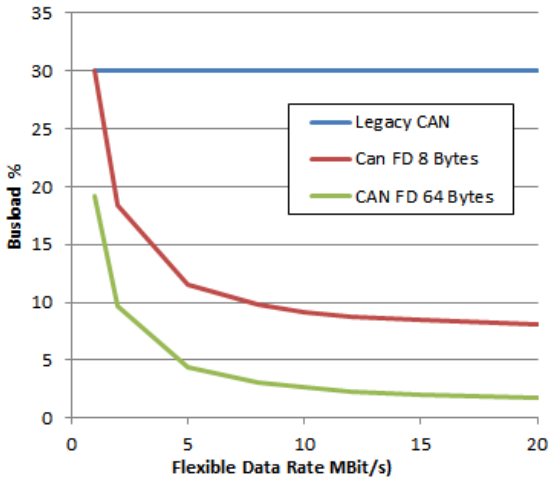

Figure 13: Effect on busload if the bytes transferred remains constant

Figure 13 plots the busload with different CAN FD parameters. With an 8 byte payload, then even relatively low flexible data rates have a significant effect on busload, falling to a third of the legacy 8 byte CAN load at an FD of 5 Mbit/s. Using 64 byte payloads further reduces the busload. This information is useful if the limit of the measure data is ECU performance. In this case the impact of XCP on the CAN bus is reduced.

The last example considered here is the possibility to measure data at faster rates using CAN FD. This use case is becoming more important with the adoption of electric and hybrid motors which require faster measure rasters then traditional combustion engine controllers. In this analysis, we will consider the effect on busload sending data in very fast rasters with different CAN FD settings.

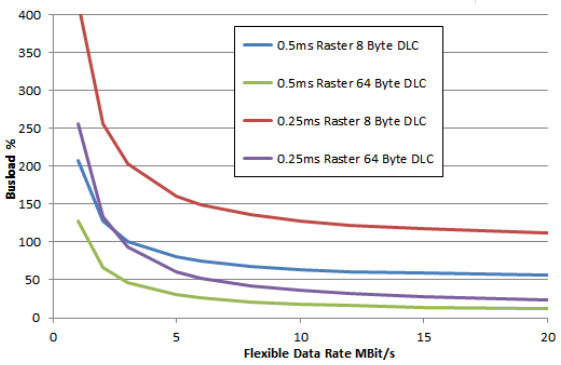

Figure 14: busload measuring data in fast rasters

In Figure 14 we show the busload required to measure 64 bytes of data in 0.5ms and 0.25ms rasters. With the legacy 8 byte CAN neither configuration is possible, with busloads over 100%. With 8 byte DLC, this measurement at 0.5ms is possible at FD rates over 2.5 Mbit/s. However measurement of 0.25ms raster is also not possible. With the adoption of 64 byte DLC it is possible to measure much faster measure rasters with acceptable busloads. So using XCP on CAN FD could result in new applications that need higher measure rates that were not possible using XCP on legacy CAN.

CAN FD effect on STIM

In terms of the XCP protocol, STIM is the same as DAQ, except it works in the opposite direction. So the master sends the slave data. However the use case behind STIM is very different to DAQ, and so the effect of CAN FD is also different. STIM is used for rapid prototyping where some or all of the calculations normally done by the ECU are done by and external processors, normal called a **R**apid **P**roto**T**yping system (RPT)

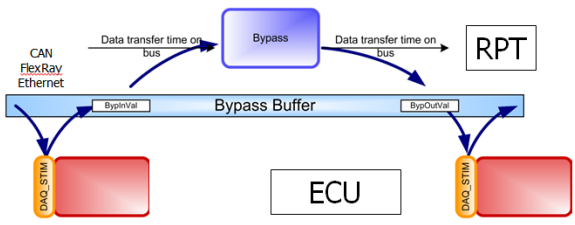

Figure 15: Basic bypass implementation

For Bypass the data has to be sent via DAQ from the ECU to the RPT, a calculation needs to occur and the data then sent back to the ECU as STIM. This sequence is shown in Figure 14. The time taken for the complete roundtrip, the so called latency, must be less than the task time the RPT is replacing. The latency is made up of the time required for the ECU to send the data, the RPT to process it, the time the bus needs to receive and send the data plus the time the ECU needs to read it back. So the bus time is only one part of this. For this reason it is difficult to judge the impact of CAN FD. Any improvements in bus transport times need to be judged in the context of the entire system before assessing any improvements that can be made with CAN FD.

Summary

Described is the CAN FD format, and XCP1.2 changes that support the CAN FD standard. This provides benefits for data downloading and flashing, which have already been publicized. However CAN FD can also bring improvements to measurement with DAQ. It has been shown that more data can be transmitted at a fixed busload, or if the same amount of data is to be sent, then the impact on the busload is reduced. In addition is it possible to send data at faster measure rates using CAN FD, which could extend the use of XCP on CAN FD buses to applications that currently cannot use CAN. The effect of CAN FD on STIM applications was considered, but because the timing of the overall system depends on more elements than just the transport layer, it is difficult to make firm conclusions of the benefit.

David Gary Hickman
ETAS GmbH
Borsigstraße 14
DE-70442 Stuttgart
Tel.: +49 711 89661 833
gary.hickman@etas.com
www.etas.com

References

[1] Bosch White Paper: CAN with Flexible Data Rate
http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can_fd
[2] ASAM XCP1.2 Standard
http://www.asam.net